

[illegible]

-3-

Synt

NTS

NTS
NTS

NTS

NTS
NTS

1994

NTS

NTS
NTS

NTS

NTS
NTS

NTS

NTS
NTS

NTS

NTS
NTS

NTS

NTS
NTS

NTS

NTS
NTS

NTS

NTS
NTS

100

N13

NTS

NTS
NTS

NTS

NTS
NTS

NTS

NT

NTS
NTS

NT

NT
NT

PI

1

1

1

1

1

2

L

```
RRRRRRRR  MM      MM      222222  CCCCCCCC  RRRRRRRR  EEEEEEEEE  AAAAAA  TTTTTTTTT  EEEEEEEEE
RRRRRRRR  MM      MM      222222  CCCCCCCC  RRRRRRRR  EEEEEEEEE  AAAAAA  TTTTTTTTT  EEEEEEEEE
RR      RR  MMMM  MMMM  22      22  CC      RRRRRRRR  RR  EE      AA      AA  TT      EE
RR      RR  MMMM  MMMM  22      22  CC      RRRRRRRR  RR  EE      AA      AA  TT      EE
RR      RR  MM      MM      22      22  CC      RRRRRRRR  RR  EE      AA      AA  TT      EE
RRRRRRRR  MM      MM      22      22  CC      RRRRRRRR  RR  EE      AA      AA  TT      EE
RRRRRRRR  MM      MM      22      22  CC      RRRRRRRR  RR  EE      AA      AA  TT      EE
RR      RR  MM      MM      22      22  CC      RR      RR  EE      AAAAAAAAAA  TT      EE
RR      RR  MM      MM      22      22  CC      RR      RR  EE      AAAAAAAAAA  TT      EE
RR      RR  MM      MM      22      22  CC      RR      RR  EE      AA      AA  TT      EE
RR      RR  MM      MM      22222222  CCCCCCCC  RR      RR  EE      AA      AA  TT      EE
RR      RR  MM      MM      22222222  CCCCCCCC  RR      RR  EE      AA      AA  TT      EE
                                         EEEEEEEEE  EEEEEEEEE  EEEEEEEEE  ....
                                         EEEEEEEEE  EEEEEEEEE  EEEEEEEEE  ....
                                         EEEEEEEEE  EEEEEEEEE  EEEEEEEEE  ....
                                         EEEEEEEEE  EEEEEEEEE  EEEEEEEEE  ....
```

```
LL      IIIIII  SSSSSSSS
LL      IIIIII  SSSSSSSS
LL      II      SS
LL      II      SS
LL      II      SS
LL      II      SS
LL      II      SSSSSS
LL      II      SSSSSS
LL      II      SS
LL      II      SS
LL      II      SS
LL      II      SS
LLLLLLLLLL  IIIIII  SSSSSSSS
LLLLLLLLLL  IIIIII  SSSSSSSS
```

(3)	103
(4)	136
(9)	414

DECLARATIONS
RMSCREATE2 - RELATIVE CREATE ROUTINE
JNL_REL_PLG - Journal the relative Prolog

[illegible]


```
0000 1 $BEGIN RM2CREATE,000,RMSRMS2,<RELATIVE-SPECIFIC CREATE>
0000 2
0000 3
0000 4 *****
0000 5 *
0000 6 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY *
0000 7 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS. *
0000 8 * ALL RIGHTS RESERVED. *
0000 9 *
0000 10 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED *
0000 11 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE *
0000 12 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER *
0000 13 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY *
0000 14 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY *
0000 15 * TRANSFERRED. *
0000 16 *
0000 17 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE *
0000 18 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT *
0000 19 * CORPORATION. *
0000 20 *
0000 21 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS *
0000 22 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL. *
0000 23 *
0000 24 *
0000 25 *****
0000 26 :
```

```
0000 28 :++
0000 29 :
0000 30 : Facility: rms32
0000 31 :
0000 32 : Abstract:
0000 33 :
0000 34 :     this routine performs the relative file
0000 35 :     organization-specific create processing.
0000 36 :
0000 37 : Environment:
0000 38 :     star processor running starlet exec.
0000 39 :
0000 40 : Author: L F Laverdure,      Creation Date: 7-DEC-1977
0000 41 :
0000 42 : Modified By:
0000 43 :
0000 44 :     V03-011 RAS0284      Ron Schaefer      30-Mar-1984
0000 45 :     Fix STV value on error paths for RMS$_RPL and RMS$_WPL errors.
0000 46 :
0000 47 :     V03-010 RAS0265      Ron Schaefer      9-Mar-1984
0000 48 :     Bump IFB$_AVLCL to count the BDB & buffer we allocate.
0000 49 :
0000 50 :     V03-009 KPL0002      Peter Lieberwirth  30-Jul-1983
0000 51 :     If AI journaling, journal the prolog.
0000 52 :
0000 53 :     V03-008 MCN0003      Maria del C. Nasr  08-Mar-1983
0000 54 :     I forgot to include $BKTDEF for MCN0002.
0000 55 :
0000 56 :     V03-007 MCN0002      Maria del C. Nasr  07-Mar-1983
0000 57 :     Use symbolic name for maximum bucket size.
0000 58 :
0000 59 :     V03-006 KBT0462      Keith B. Thompson  13-Jan-1983
0000 60 :     Allocate a bdb and buffer to read in prologue
0000 61 :
0000 62 :     V03-005 MCN0001      Maria del C. Nasr  16-Dec-1982
0000 63 :     Maximum number of blocks per bucket was increased from
0000 64 :     32 to 127.
0000 65 :
0000 66 :     V03-004 KBT0332      Keith B. Thompson  10-Sep-1982
0000 67 :     Remove $FRBDEF
0000 68 :
0000 69 :     V03-003 KBT0132      Keith B. Thompson  20-Aug-1982
0000 70 :     Reorganize psects
0000 71 :
0000 72 :     V03-002 KBT0116      Keith B. Thompson  6-Aug-1982
0000 73 :     Remove ref. to set_sifb_ptr
0000 74 :
0000 75 :     V03-001 KBT0097      Keith B. Thompson  13-Jul-1982
0000 76 :     Clean up psects
0000 77 :
0000 78 :     V02-017 CDS0012      C Saether      5-Feb-1982
0000 79 :     Back out V02-016. GBC now in record attributes.
0000 80 :
0000 81 :     V02-016 CDS0011      C Saether      3-Jan-1982
0000 82 :     Store GBC field from FAB to plg.
0000 83 :
0000 84 :     V02-015 CDS0010      C Saether      25-Aug-1981
```

0000	85	:	Replace call to RMSALLOC_BCB with RMSALBLB.
0000	86	:	
0000	87	:	
0000	88	:	V02-014 RAS0028 Ron Schaefer 20-Aug-1981
0000	89	:	Change FABSC_STM11 to FABSC_STM.
0000	90	:	
0000	91	:	V02-013 RAS0015 Ron Schaefer 7-Jul-1981
0000	92	:	Correct record format check for stream format files.
0000	93	:	
0000	94	:	V02-012 KPL0001 Peter Lieberwirth 24-Jul-1981
0000	95	:	Fix broken branches.
0000	96	:	
0000	97	:	V02-011 CDS0012 C SAETHER 28-Aug-1980 16:00
0000	98	:	Fix sense of test in V009.
0000	99	--	
0000	100	:	
0000	101	:	


```

0000 103      .SBTTL DECLARATIONS
0000 104
0000 105 :
0000 106 : Include Files:
0000 107 :
0000 108 :
0000 109 :
0000 110 : Macros:
0000 111 :
0000 112 :
0000 113      $FABDEF
0000 114      $IFBDEF
0000 115      $BKTDEF
0000 116      $CSHDEF
0000 117      $DEVDEF
0000 118      $BDBDEF
0000 119      $PLGDEF
0000 120      $RLSDEF
0000 121      $RMSDEF
0000 122      $RJRDEF
0000 123      $CJFDEF
0000 124
0000 125 :
0000 126 : Equated Symbols:
0000 127 :
0000 128
00000020 0000 129      FOP=FAB$L_FOP*8      ; bit offset to fop
0000 130
0000 131 :
0000 132 : Own Storage:
0000 133 :
0000 134

```

```
0000 136 .SBTTL RM$CREATE2 - RELATIVE CREATE ROUTINE
0000 137
0000 138 :++
0000 139
0000 140 RM$CREATE2
0000 141
0000 142 RM$CREATE2 -
0000 143
0000 144 this routine performs all of the file create
0000 145 functions that are specific to the relative
0000 146 file organization, including:
0000 147
0000 148 1. checking that sharing has not been specified in such a way
0000 149 that inter-process record locking is required.
0000 150 2. checking that device is a disk if not bio mode
0000 151 3. checking that record format is not undefined or stream
0000 152 4. checking that bucket size and maximum record size are compatible
0000 153 5. verifying maximum record number
0000 154 6. checking xab chain validity
0000 155 7. calling the common create routine
0000 156 8. locking the prolog, initial formatting of the data buckets to zeroes
0000 157 9. initializing and unlocking the prolog
0000 158
0000 159 Calling sequence:
0000 160
0000 161 entered via case branch from rm$open
0000 162 returns by jumping to rm$createxit
0000 163
0000 164 Input Parameters:
0000 165
0000 166 r11 impure area address
0000 167 r10 fwa address
0000 168 r9 ifab address
0000 169 r8 fab address
0000 170
0000 171 Implicit Inputs:
0000 172
0000 173 the contents of the fab, ifab, & fwa.
0000 174
0000 175 Output Parameters:
0000 176
0000 177 r0 status code
0000 178 r1-r7 destroyed
0000 179
0000 180 Implicit Outputs:
0000 181
0000 182 various fields in the ifab & fab are initialized.
0000 183
0000 184 Completion Codes:
0000 185
0000 186 standard rms
0000 187
0000 188 Side Effects:
0000 189
0000 190 none
0000 191
0000 192 :--
```


RM2CREATE
V04-000

RELATIVE-SPECIFIC CREATE
RM\$CREATE2 - RELATIVE CREATE ROUTINE

B 2

16-SEP-1984 01:01:32 VAX/VMS Macro V04-00
5-SEP-1984 16:24:00 [RMS.SRC]RM2CREATE.MAR;1

Page 6
(4)

0000 193

RM2
V04

```
0000 195
0000 196 :
0000 197 : code to handle error conditons.
0000 198 : (note: this is not the entry point for the rm$create2 routine.)
0000 199 :
0000 200
0000 201 ERRDEV:
0000 202 RMSERR DEV ; device not disk
FFF8' 31 0005 203 ERRXIT: BRW RM$CREATEEXIT ; go clean up
0008 204
FFF5' 31 0008 205 ERRRFM: BRW RM$CRE_ERRRFM ; rfm = udf or > vfc
000B 206
FFF2' 31 000B 207 ERRMRS: BRW RM$CRE_ERRMRS ; mrs < or = 0
000E 208
000E 209 ERRBKS:
000E 210 RMSERR BKS ; bks > BKT$C_MAXBKTSIZ or < cell size
F0 11 0013 211 BRB ERRXIT ; go clean up
0015 212
0015 213 ERRMRN:
0015 214 RMSERR MRN ; mrn < 0
E9 11 001A 215 BRB ERRXIT ; go clean up
001C 216
```

```
001C 218
001C 219 :++
001C 220 : entry point for relative-specific create
001C 221 :
001C 222 :--
001C 223
001C 224 RM$CREATE2::
001C 225     $TSTPT CREATE2
0022 226
0022 227 :
0022 228 : check that device is disk
0022 229 :
0022 230 :
22 A9 05 E0 0022 231     BBS     #IFB$V_BIO,IFB$B_FAC(R9),-
        04          0026 232     $S          ; allow bio on any dev
69 1C E1 0027 233     BBC     #DEV$V_RND,IFB$L_PRIM_DEV(R9),-
        D5          002A 234     ERRDEV      ; branch if not disk
002B 235
002B 236 :
002B 237 : handle allocation request, if any
002B 238 :
002B 239 :
        FFD2' 30 002B 240 $S:     BSBW     RM$SETALLOC      ; handle allocation xab and
002E 241                                ; set deq and rtdeq
002E 242 ERRXIT1:
D4 50 E9 002E 243     BLBC     R0,ERRXIT      ; get out on error
10 A8 D5 0031 244     TSTL     FAB$L_ALQ(R8)      ; any initial allocation?
        03 12 0034 245     BNEQ     10$          ; branch if yes
10 A8 D6 0036 246     INCL     FAB$L_ALQ(R8)      ; no - need 1 block for prolog
0039 247
0039 248 :
0039 249 : check rfm and mrs parameters
0039 250 :
0039 251 : assume rfm already checked for gtr than maxrfm
0039 252 :
0039 253 :
0039 254     ASSUME   FAB$C_UDF      EQ      0
0039 255
50 A9 95 0039 256 10$:     TSTB     IFB$B_RFMORG(R9)      ; is rfm undefined?
        CA 13 003C 257     BEQL     ERRRFM      ; branch if yes
003E 258
003E 259     ASSUME   FAB$C_STM      GT      FAB$C_VFC
003E 260
50 A9 91 003E 261     CMPB     IFB$B_RFMORG(R9),-
        04          0041 262     #FAB$C_STM      ; is rfm stream?
        C4 1E 0042 263     BGEQU     ERRRFM      ; branch if yes
0044 264
S2 A9 36 A8 B0 0044 265     MOVW     FAB$W_MRS(R8),IFB$W_LRL(R9)-
        C0 15 0049 266                                ; set lrl from fab mrs
0049 267     BLEQ     ERRMRS      ; branch if not > 0
004B 268
004B 269 :
004B 270 : compute cell size
004B 271 :
004B 272 :
50 01 36 A8 A1 004B 273     ADDW3    FAB$W_MRS(R8),#1,R0      ; add in delete ctrl byte
50 A9 91 0050 274     CMPB     IFB$B_RFMORG(R9),-
```



```
01 0053 275      #FAB$C_FIX      : fixed rec len?
0A 13 0054 276      BEQL 30$      : branch if yes
51 50 02 A0 0056 277      ADDW2 #2,R0      : add in record length field
51 5F A9 9A 0059 278      MOVZBL IFB$B_FSZ(R9),R1      : get fsz
50 51 A0 005D 279      ADDW2 R1,R0      : and add in giving tot. size
0060 280
0060 281
0060 282      : check cell size against bks
0060 283
0060 284
51 3E AB 9A 0060 285 30$: MOVZBL FAB$B_BKS(R8),R1      : copy bucket size from fab
OC 12 0064 286      BNEQ 40$      : branch if speced
0065 287
0066 288
0066 289      : default bucket size to min.
0066 290      : required to contain 1 record
0066 291
0066 292
51 50 0200 50 B7 0066 293      DECW R0      : round down
8F A7 0068 294      DIVW3 #512,R0,R1      : get # blks - 1 for 1 record
51 B6 006E 295      INCW R1      : get # blks for 1 record
50 B6 0070 296      INCW R0      : restore cell size
5E A9 51 90 0072 297 40$: MOVB R1,IFB$B_BKS(R9)      : copy bucket size to ifab
3F 51 91 0076 298      CMPB R1,#BKT$C_MAXBKTSIZ      : in range?
93 1A 0079 299      BGTRU ERBKS      : branch if not
51 51 09 78 007B 300      ASHL #9,R1,R1      : compute bucket size in bytes
51 51 50 B1 007F 301      CMPW R0,R1      : cell size < or = bucket size?
8A 1A 0082 302      BGTRU ERBKS      : branch if not
0084 303      : set mrn value
00AC C9 38 AB D0 0084 304      MOVL FAB$L_MRN(R8),IFB$L_MRN(R9)
008A 305      : set mrn from fab
OB 14 008A 306      BGTR 50$      : branch if > 0
87 19 008C 307      BLSS ERRMRN      : error if < 0
00AC C9 7FFFFFFF 8F D0 008E 308      MOVL #^X7FFFFFFF,IFB$L_MRN(R9)
0097 309      : default to max. pos #
0097 310
0097 311
0097 312      : go do create.
0097 313      : (note: this may be a 'create if', in which case return will be
0097 314      : made to rms0open if actually opened rather than created.)
0097 315
0097 316
FF66' 30 0097 317 50$: BSBW RMS$CREATECOM      : do common create
91 50 E9 009A 318      BLBC R0,ERRXIT1      : get out on error
50 DD 009D 319      PUSHL R0      : save status code
009F 320
009F 321
009F 322      : file has been created.
009F 323      : allocate a lock bdb and bcb and lock the prolog.
009F 324
009F 325
22 5A 59 D0 009F 326      MOVL R9,R10      : set r10 to ifab addr
A9 05 E1 00A2 327      BBC #IFB$V_BIO,IFB$B_FAC(R9)      :
0096 31 00A6 328      : continue unless block i/o
55 0200 8F 3C 00AA 329      BRW EXIT      : avoid formatting for block io
FF4E' 30 00AF 330 52$: MOVZWL #512,R5      : ask for 1 block to read prologue
BSBW RMS$ALDBUF      : get bdb and buffer
```

```

      46 50 E9 00B2 332      BLBC      R0, 70$      ; Branch on error.
      00B4 C9 B6 00B3 333      INCW      IFBSW_AVLCL(R9)      ; count BDB & buffer
06 6A 33 E0 00B9 334      BBS      #IFBSV_NORECLK,(R10),55$ ; Branch if not locking.
      FF40' 30 00BD 335      BSBW      RMSALBCLB      ; Get a lock BLB.
      38 50 E9 00C0 336      BLBC      R0, 70$      ; Branch on error.
      00C3 337 55$: $CACHE VBN=#1,-
      00C3 338      SIZE=#0,-
      00C3 339      FLAGS=<LOCK,NOREAD,NOBUFFER>
      2A 50 E9 00CE 340      BLBC      R0,70$      ; branch on error
      00D1 341
      00D1 342
      00D1 343
      00D1 344
      00D1 345
      00D1 346      MOVL      #2,IFBSL_DVBN(R9)      ; set first data vbn
      51 02 D0 00D6 347      MOVL      #2,R1      ; 1st block for zeroing
56 70 A9 01 C1 00D9 348      ADDL3     #1,IFBSL_HBK(R9),R6 ; compute eof block
      74 A9 56 D0 00DE 349      MOVL      R6,IFBSL_EBK(R9) ; save it
      02 56 D1 00E2 350      CMPL      R6,#2      ; eof in vbn 2?
      06 13 00E5 351      BEQL      60$      ; branch if yes (no need to zero)
      FF16' 30 00E7 352      BSBW      RMSFMT_BKT2 ; format (zero) data buckets
      56 50 E9 00EA 353      BLBC      R0,RLNERR ; branch on error
      00ED 354
      00ED 355
      00ED 356
      00ED 357
      00ED 358
      00ED 359 60$: $CACHE VBN=#1,-
      00ED 360      SIZE=#512,-
      00ED 361      FLAGS=<LOCK,NOREAD> ; get buffer for prolog
      45 50 E9 00FB 362 70$: BLBC      R0,ERRBUG ; branch on error
      00 6E 00 2C 0100 363      PUSHR     #^M<R4,R5> ; save bdb and buffer addr
      30 BA 0108 364      MOVCS     #0,(SP),#0,#512,(R5) ; zero buffer
      74 A5 01 B0 010A 365      POPR      #^M<R4,R5> ; restore bdb and buffer addr
      010E 366      MOVW      #PLGSC_VER_NO,PLGSW_VER_NO(R5) ; set version #
      70 A5 56 D0 010E 367      MOVL      R6,PLGSL_EOF(R5) ; and eof vbn
68 A5 00B0 C9 B0 0112 368      MOVW      IFBSL_DVBN(R9),PLGSW_DVBN(R5) ; and first data vbn
6C A5 00AC C9 D0 0118 369      MOVL      IFBSL_MRN(R9),PLGSL_MRN(R5) ; and max record number
      FEDF' 30 011E 370      BSBW      RMSMAKSUM ; calculate and set checksum
      0A A4 03 88 0121 371      BISB2     #BDBSM_DRT!BDBSM_VAL,BDBSM_FLGS(R4) ; say valid and dirty
      53 02 D0 0125 372      MOVL      #RLSM_WRT_THRU,R3 ; cause immediate write
      7E 55 D0 0128 373      MOVL      R5,-(SP) ; protect PLG address from RELEASE
      FED2' 30 012B 374      BSBW      RMSRELEASE ; release prolog
      55 8E D0 012E 375      MOVL      (SP)+,R5 ; restore PLG address
      24 50 E9 0131 376      BLBC      R0,RLSERR ; branch on error
      0134 377
      0134 378
      0134 379
      0134 380
      0134 381
      0134 382
      0134 383
      0134 384
      0134 385
      06 00A0 C9 03 E1 0134 386      BBC      #IFBSV_AI,IFBSB_JNLFLG(R9),EXIT ; skip if not AI journaling
      003B 30 013A 387      BSBW      JNL_REC_PLG ; journal the prolog
      2D 50 E9 013D 388      BLBC      R0,ERRJNL ; branch on error
```

```
FEBD' 31 0140 389 EXIT: BRW RMSCREATEXIT1 ; finish up create
          0143 390
          0143 391 :: handle errors
          0143 392 ::
          0143 393 ::
          0143 394
          0143 395 ERRBUG:
          0143 396 RLNERR: ; failed zero data buckets
          50 DD 0143 397 PUSHL RO ; store status
          0145 398 $CACHE VBN=#1,-
          0145 399 SIZE=#0,-
          0145 400 ERR=EXIT ; re-get prolog bdb
00000000'EF 16 0150 401 JSB RMSRLNERR ; unlock prolog
          E8 11 0156 402 BRB EXIT ; and get out
          0158 403
          OC AB D5 0158 404 RLSERR: TSTL FAB$L_STV(R8) ; do we have an stv?
          09 12 015B 405 BNEQ 10$ ; okay use it
OC AB 6E 00001000 8F C9 015D 406 BISL3 #^X1000,(SP),FAB$L_STV(R8) ; else set the RMS error there
          D3 11 0166 407 10$: RMSERR WPL,(SP) ; prolog write error
          016B 408 BRB EXIT ; go clean up
          016D 409
          016D 410 ERRJNL: RMSERR CJF,(SP) ; journal write error
          OC AB 50 D0 0172 411 MOVL RO,FAB$L_STV(R8) ; save CJF status where user can find it
          C8 11 0176 412 BRB EXIT ; go clean up
```



```
0178 414 .SUBTITLE JNL_REL_PLG - Journal the relative Prolog
0178 415 :++
0178 416 JNL_REL_PLG
0178 417 :
0178 418 This routine writes the prolog as a block entry to the AI journal.
0178 419 :
0178 420 Inputs:
0178 421 :
0178 422 r9 IFAB
0178 423 r5 PLG
0178 424 :
0178 425 Outputs:
0178 426 :
0178 427 r0 status
0178 428 :
0178 429 PROLOG witten to the journal.
0178 430 :
0178 431 :--
0178 432
0178 433 JNL_REL_PLG:
0178 434 :
53 30 A9 D0 0178 435 MOVL IFB$L_JNLBDB(R9),R3 ; get address of BDB/Buffer
52 18 A3 D0 017C 436 MOVL BDB$L_ADDR(R3),R2 ; get RJR address
0180 437 :
0180 438 :
0180 439 : Set up the common RJR overhead.
0180 440 :
03 A2 03 90 0180 441 MOVB #RJR$C_BLOCK,RJR$B_ENTRY_TYPE(R2) ; block IO
04 A2 01 90 0184 442 MOVB #RJR$C_REL,RJR$B_ORG(R2) ; file organization
05 A2 1E 90 0188 443 MOVB #RJR$C_WRITE,RJR$B_OPER(R2) ; operation
018C 444 :
018C 445 :
018C 446 : Set up the block IO entry.
018C 447 :
40 A2 3C A2 01 D0 018C 448 MOVL #1,RJR$L_BLOCK_VBN(R2) ; PROLOG is VBN 1
00000200 8F D0 0190 449 MOVL #512,RJR$L_BLOCK_SIZE(R2) ; size of PROLOG is 512 bytes
3C BB 0198 450 PUSHR #M<R2,R3,R4,R5> ; save MOV C3 regs
44 A2 64 0200 8F 28 019A 451 MOV C3 #512,(R4),RJR$T_BLOCK(R2) ; copy the prolog
3C BA 01A1 452 POPR #M<R2,R3,R4,R5> ; restore MOV C3 regs
01A3 453 :
01A3 454 :
01A3 455 : Set up the WRTJNL call parameters.
01A3 456 :
7E 53 D0 01A3 457 MOVL R3,-(SP) ; JNLBDB address
7E 03 D0 01A6 458 MOVL #CJF$ AI,-(SP) ; AI journaling
00000000 EF 16 01A9 459 JSB RMS$WRTJNL ; write entry to journal
5E 08 C0 01AF 460 ADDL2 #8,SP ; pop parameters off stack
05 01B2 462 RSB ; return WRTJNL status to caller
01B3 463 :
01B3 464 .END
```

Variable	Value	Mode	Priority
\$\$PSECT_EP	= 00000000		
\$\$TMP	= 00000005		
\$\$RMSTEST	= 0000001A		
\$\$RMS_PBUGCHK	= 00000010		
\$\$RMS_TBUGCHK	= 00000008		
\$\$RMS_UMODE	= 00000004		
BDB\$B_FLGS	= 0000000A		
BDB\$B_ADDR	= 00000018		
BDB\$B_DRT	= 00000002		
BDB\$B_VAL	= 00000001		
BKT\$C_MAXBKTSIZ	= 0000003F		
CJFS_A1	= 00000003		
CSH\$M_LOCK	= 00000001		
CSH\$M_NOBUFFER	= 00000008		
CSH\$M_NOREAD	= 00000004		
DEV\$V_RND	= 0000001C		
ERRBK5	0000000E	R	01
ERRBUG	00000143	R	01
ERRDEV	00000000	R	01
ERRJNL	0000016D	R	01
ERRMRN	00000015	R	01
ERRMRS	0000000B	R	01
ERRRFM	00000008	R	01
ERRXIT	00000005	R	01
ERRXIT1	0000002E	R	01
EXIT	00000140	R	01
FAB\$B_BKS	= 0000003E		
FAB\$C_FIX	= 00000001		
FAB\$C_STM	= 00000004		
FAB\$C_UDF	= 00000000		
FAB\$C_VFC	= 00000003		
FAB\$B_ALQ	= 00000000		
FAB\$B_FOP	= 00000004		
FAB\$B_MRN	= 00000038		
FAB\$B_STV	= 0000000C		
FAB\$W_MRS	= 00000036		
FOP	= 00000020		
IFB\$B_BKS	= 0000005E		
IFB\$B_FAC	= 00000022		
IFB\$B_FSZ	= 0000005F		
IFB\$B_JNLFLG	= 000000A0		
IFB\$B_RFMORG	= 00000050		
IFB\$B_DVBN	= 000000B0		
IFB\$B_EBK	= 00000074		
IFB\$B_HBK	= 00000070		
IFB\$B_JNLBDB	= 00000030		
IFB\$B_MRN	= 000000AC		
IFB\$B_PRIM_DEV	= 00000000		
IFB\$V_A1	= 00000003		
IFB\$V_BIO	= 00000005		
IFB\$V_NORECLK	= 00000033		
IFB\$W_AVLCL	= 00000084		
IFB\$W_LRL	= 00000052		
JNL_REL_PLG	00000178	R	01
PIOSA_TRACE	*****	X	01
PLG\$C_VER_NO	= 00000001		
PLG\$B_EOF	= 00000070		

PLG\$\$_MRN	=	0000006C		
PLG\$\$_DVBN	=	00000068		
PLG\$\$_VER_NO	=	00000074		
RJR\$\$_ENTRY_TYPE	=	00000003		
RJR\$\$_OPER	=	00000005		
RJR\$\$_ORG	=	00000004		
RJR\$\$_BLOCK	=	00000003		
RJR\$\$_REL	=	00000001		
RJR\$\$_BLOCK_SIZE	=	00000040		
RJR\$\$_BLOCK_VBN	=	0000003C		
RJR\$\$_BLOCK	=	00000044		
RJR\$\$_WRITE	=	0000001E		
RLNERR		00000143	R	01
RLS\$\$_WRT_THRU	=	00000002		
RLSERR		00000158	R	01
RM\$\$_ALBLB	*****		X	01
RM\$\$_ALDBUF	*****		X	01
RM\$\$_CACHE	*****		X	01
RM\$\$_CREATE2	0000001C	RG		01
RM\$\$_CREATECOM	*****		X	01
RM\$\$_CREATEEXIT	*****		X	01
RM\$\$_CREATEEXIT1	*****		X	01
RM\$\$_CRE_ERRMRS	*****		X	01
RM\$\$_CRE_ERRRFM	*****		X	01
RM\$\$_FMT_BKT2	*****		X	01
RM\$\$_MAKSUM	*****		X	01
RM\$\$_RELEASE	*****		X	01
RM\$\$_RLNERR	*****		X	01
RM\$\$_SETALLOC	*****		X	01
RM\$\$_WRTJNL	*****		X	01
RM\$\$_BKS	=	0001841C		
RM\$\$_CJF	=	0001C164		
RM\$\$_DEV	=	000184C4		
RM\$\$_MRN	=	000185CC		
RM\$\$_WPL	=	0001C11C		
TPT\$\$_CREATE2	*****		X	01

[illegible]

PSE

RMS
SAE

+-----+
! Psect synopsis !
+-----+

PSECT name	Allocation	PSECT No.	Attributes
ABS	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
RMSRMS2	000001B3 (435.)	01 (1.)	PIC USR CON REL GBL NOSHR EXE RD NOWRT NOVEC BYTE
SABSS	00000000 (0.)	02 (2.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE

+-----+
! Performance indicators !
+-----+

Phase	Page faults	CPU Time	Elapsed Time
Initialization	33	00:00:00.07	00:00:00.68
Command processing	118	00:00:00.72	00:00:06.05
Pass 1	324	00:00:11.06	00:00:28.57
Symbol table sort	0	00:00:01.42	00:00:02.17
Pass 2	88	00:00:02.16	00:00:05.42
Symbol table output	12	00:00:00.15	00:00:00.44
Psect synopsis output	2	00:00:00.02	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	579	00:00:15.60	00:00:43.35

The working set limit was 1350 pages.
60191 bytes (118 pages) of virtual memory were used to buffer the intermediate code.
There were 60 pages of symbol table space allocated to hold 1116 non-local and 10 local symbols.
464 source lines were read in Pass 1, producing 14 object records in Pass 2.
26 pages of virtual memory were used to define 25 macros.

+-----+
! Macro library statistics !
+-----+

Macro library name	Macros defined
\$255\$DUA28:[RMS.OBJ]RMS.MLB;1	15
\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	0
\$255\$DUA28:[SYSLIB]STARLET.MLB;2	6
TOTALS (all libraries)	21

1255 GEIS were required to define 21 macros.
There were no errors, warnings or information messages.
MACRO/LIS=LIS\$:RM2CREATE/OBJ=OBJ\$:RM2CREATE MSRC\$:RM2CREATE/UPDATE=(ENH\$:RM2CREATE)+EXECML\$/LIB+LIB\$:RMS/LIB

RM2
VAX

Pha

Ini
Com
Pas
Sym
Pas
Sym
Pse
Cro
Ass

The
467
The
372
22

Mac

\$2
\$2
\$2
TOT

997

The
MAC

0323

AH-BT13A-SE
VAX/VMS V4.0

**DIGITAL
CONFIDENTIAL**

EQUIPMENT
NTIAL AND

CORPORATION
PROPRIETARY